

FFFFFF	000000000	RRRRRRRRRRRR	RRRRRRRRRRRR	TTTTTTTTTTTTT	LLL	
FFFF	000000000	RRRRRRRRRRRR	RRRRRRRRRRRR	TTTTTTTTTTTTT	LLL	
FFFF	000000000	RRRRRRRRRRRR	RRRRRRRRRRRR	TTTTTTTTTTTTT	LLL	
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFFF	000	000	RRRRRRRRRRRR	RRRRRRRRRRRR	TTT	LLL
FFFF	000	000	RRRRRRRRRRRR	RRRRRRRRRRRR	TTT	LLL
FFFF	000	000	RRRRRRRRRRRR	RRRRRRRRRRRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000	000	RRR	RRR	TTT	LLL
FFF	000000000	RRR	RRR	RRR	TTT	LLLLLLLLLLLL
FFF	000000000	RRR	RRR	RRR	TTT	LLLLLLLLLLLL
FFF	000000000	RRR	RRR	RRR	TTT	LLLLLLLLLLLL

The diagram illustrates a 2D convolution operation. The input layer (bottom) consists of two rows of 8 'L's each. The kernel (middle) is a 5x5 matrix of 'I's. The output layer (top) shows the result of the convolution, with 8 'S's in the first row and 16 'S's in the second row. The 'S's are aligned with the 'L's in the input layer, indicating the receptive field of each output unit.

```

1 0001 0 MODULE FOR$ERROR (XTITLE 'Internal FORTRAN error handling module'
2 0002 0 IDENT = '1-022'          ! File: FORERROR.B32 Edit: SBL1022
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1 ****
6 0006 1 *
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 ****
28 0028 1 !
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: FORTRAN support library
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains the error handlers needed by
36 0036 1 the common OTS for handling FORTRAN errors. In particular
37 0037 1 there is a handler for errors in OPEN/CLOSE where ERR=
38 0038 1 means error return to caller rather than a transfer.
39 0039 1 A second handler (FOR$ERR_END_HND is provided
40 0040 1 for I/O statements where the optional ERR= and END=
41 0041 1 constructs require a transfer of control to the
42 0042 1 user program rather than an error return.
43 0043 1 A third handler, FOR$IOSTAT_HND is for auxilliary I/O statements
44 0044 1 which either unwind with R0 containing an IOSTAT value or
45 0045 1 resignal.
46 0046 1 An argument specifies the cleanup to be performed if UNWIND occurs.
47 0047 1
48 0048 1 ENVIRONMENT: User mode, AST level or not or mixed.
49 0049 1 Note: this module is both shared (with no entry vectors) and non-shared
50 0050 1 if FORTRAN compatibility routines call.
51 0051 1
52 0052 1 AUTHOR: Thomas N. Hastings, CREATION DATE: 03-Jun-77
53 0053 1
54 0054 1 MODIFIED BY:
55 0055 1
56 0056 1 Thomas N. Hastings, 03-Jun-77: VERSION 01
57 0057 1 Steven B. Lionel, VAX/VMS V2.0

```

FOR\$ERROR  
1-022

Internal FORTRAN error handling module

M 4  
16-Sep-1984 00:20:31  
14-Sep-1984 12:31:54  
VAX-11 Bliss-32 V4.0-742  
[FORRTL.SRC]FORERROR.B32;1

Page 2  
(1)

58 0058 1 | [Previous edit history deleted. SBL 30-Sep-1982]  
59 0059 1 | 1-019 - Look at FAO\_COUNT in signal list to see where USER\_PC is. SBL 10-NOV-1980  
60 0060 1 | 1-020 - Reset RAB\$L\_UBF and RAB\$W\_USZ in CLEANUP\_LUB. JAW-08-Jun-1981  
61 0061 1 | 1-021 - Change OTSS\$ data structure references to FORSS. SBL 30-Sep-1982  
62 0062 1 | 1-022 - Look at FAB\$WIFI instead of LUB\$WIFI in CLEANUP\_LUB. QAR #1229.  
63 0063 1 | SBL 7-Mar-1984  
64 0064 1 | --  
65 0065 1 |

FOR  
1-02

```

67 0066 1 ! PROLOGUE FILE:
68 0067 1 !
69 0068 1 !
70 0069 1 !
71 0070 1 REQUIRE 'RTLIN:FORPROLOG';           ! FORTRAN definitions
72 0136 1 !
73 0137 1 !
74 0138 1 ! TABLE OF CONTENTS:
75 0139 1 !
76 0140 1 !
77 0141 1 FORWARD ROUTINE
78 0142 1 FOR$ERR_OPECLO,           ! Error handler for OPEN/CLOSE
79 0143 1 FOR$ERR_ENDHND,          ! ERR=/END= handler for I/O statements
80 0144 1 FOR$IOSTAT_HND,          ! IOSTAT only handler
81 0145 1 FOR$IO_IN_PROG,          ! I/O in progress handler
82 0146 1 CLEANUP_LUB : NOVALUE;   ! Perform appropriate LUB cleanup if UNWIND.
83 0147 1 !
84 0148 1 !
85 0149 1 !
86 0150 1 ! EQUATED SYMBOLS:
87 0151 1 !
88 0152 1 !     NONE
89 0153 1 !
90 0154 1 ! OWN STORAGE:
91 0155 1 !
92 0156 1 !     NONE
93 0157 1 !
94 0158 1 ! EXTERNAL REFERENCES:
95 0159 1 !
96 0160 1 +
97 0161 1 ! MAINTENANCE NOTE: Since this module is called by FORTRAN compatibility
98 0162 1 ! routines which are un-shared and the entry points are not vectored,
99 0163 1 ! a separate copy of this module is linked with the user program when
100 0164 1 ! the user calls a FORTRAN compatibility routine. In order to prevent
101 0165 1 ! data truncation errors from the linker, all external references are
102 0166 1 ! of addressing mode general (rather than word displacement) even for
103 0167 1 ! the same PSELECT.
104 0168 1 !
105 0169 1 !
106 0170 1 ! EXTERNAL ROUTINE
107 0171 1 !     FOR$CB_GET : JSB_CB_GET NOVALUE,           ! Get current LUB/ISB/RAB
108 0172 1 !                                         Note: this non-shared routine is loaded if
109 0173 1 !                                         compatibility routines call, so can't reference
110 0174 1 !                                         FOR$A_CUR_LUB directly.
111 0175 1 !     FOR$CB_POP : JSB_CB_POP NOVALUE,          ! Pop current LUB/ISB/RAB
112 0176 1 !                                         as specified by CCB.
113 0177 1 !     FOR$FP_MATCH : CALL_CCB NOVALUE,          ! Match FP in ISB chain
114 0178 1 !     FOR$FREE_VM,                           ! Free virtual memory
115 0179 1 !     FOR$CLOSE_FILE,                         ! RMS Close a file
116 0180 1 !     FOR$$SIG_FATINT : NOVALUE,            ! SIGNAL_STOP OTSS_FATINTERR
117 0181 1 !     FOR$$SIG_DATCOR : NOVALUE,            ! SIGNAL_STOP OTSS_INTDATCOR
118 0182 1 !                                         (FATAL INTERNAL ERROR IN RUN-TIME LIBRARY)
119 0183 1 !     LIBSSIG_TO_RET;                      ! convert a SIGNAL to error return
120 0184 1 !                                         ! to caller of establisher with R0 set to signal value.
121 0185 1 !

```

```

123 0186 1 GLOBAL ROUTINE FOR$ERR_OPECLO (
124 0187 1     SIG_ARGS_ADR,
125 0188 1     MCH_ARGS_ADR,
126 0189 1     ENB_ARGS_ADR)
127 0190 1     =
128 0191 1
129 0192 1     ++
130 0193 1     FUNCTIONAL DESCRIPTION:
131 0194 1
132 0195 1     FOR$ERR_OPECLO is an error condition handler established by
133 0196 1     the OPEN and CLOSE statement procedures. If the user specified
134 0197 1     an ERR= keyword parameter, the handler unwinds the stack after
135 0198 1     storing the signaled error condition in the saved image of R0.
136 0199 1     Otherwise, FOR$ERR_OPECLO just resignals by simply returning
137 0200 1     SSS_RESIGNAL (to CHF).
138 0201 1     If and when an UNWIND occurs, the ENABLE arg UNWIND_ACT_ADR
139 0202 1     specifies whether the LUB/ISB/RAB is to be pop, returned, or no-opped.
140 0203 1     It is not popped if it had not yet been pushed as indicated
141 0204 1     by the ENABLE arg UNWIND_ACT_ADR.
142 0205 1
143 0206 1     If ERR= and IOSTAT were both specified, then the returned
144 0207 1     value is the FORTRAN small integer error code.
145 0208 1
146 0209 1     FORMAL PARAMETERS:
147 0210 1
148 0211 1     SIG-ARG-ADR
149 0212 1     SIG_ARGS_ADR.r1.ra      Adr. of Signal arg list
150 0213 1     MCH_ARGS_ADR.r1.ra      Adr. of mechanism arg list
151 0214 1     ENB_ARGS_ADR.r1.ra      Adr. of ENABLE arg list which contains:
152 0215 1         ENAB[LE_COUNT.rbu.v  No. of longword following in ENABLE arg list
153 0216 1         UNWIND_ACT_ADR.r1.r  Adr. of longword containing UNWIND action code.
154 0217 1         Any of FOR$K_UNWINDNOP, FOR$K_UNWINDPOP,
155 0218 1         FOR$K_UNWINDRET.
156 0219 1     [OPECLO_ADR.r1u.ra] Optional adr. of canonical array of OPEN or CLOSE keyword
157 0220 1     parameters after the encoded user parameter
158 0221 1     list has been scanned and expanded into it.
159 0222 1     Symbolic offsets into ENB_ARGS_ADR[1,OPEN$K_name] are of the
160 0223 1     form OPEN$K_name as defined in FOROPN REQUIRE file.
161 0224 1     If omitted, assume no ERR= (DEFINE FILE, REWIND, etc)
162 0225 1
163 0226 1     IMPLICIT INPUTS:
164 0227 1
165 0228 1     FOR$SA_CUR_LUB      Adr. of current LUB/ISB/RAB or 0
166 0229 1
167 0230 1
168 0231 1
169 0232 1     IMPLICIT OUTPUTS:
170 0233 1
171 0234 1     SIG_ARGS_ADR[SIG$USER_PC]  Set to user call PC to RTL
172 0235 1
173 0236 1     COMPLETION CODES:
174 0237 1
175 0238 1     SSS_RESIGNAL if no ERR= was specified
176 0239 1     SSS_NORMAL if ERR= was specified (ignored by CHF on UNWIND)
177 0240 1
178 0241 1     SIDE EFFECTS:
179 0242 1

```

```
180 0243 1 ! If the user has specified ERR=, the stack is unwound to the
181 0244 1 ! caller of the establisher (i.e., the user program) with the save image
182 0245 1 ! of R0 set to the error status.
183 0246 1 ! If no ERR= was specified, the error condition is signaled.
184 0247 1 ! If UNWIND call, the current LUB/ISB/RAB may be popped or returned.
185 0248 1 !--  
186 0249 1 BEGIN  
187 0250 2 BUILTIN  
188 0251 2 CALLG,  
189 0252 2 AP;  
190 0253 2  
191 0254 2  
192 0255 2  
193 0256 2 LITERAL  
194 0257 2 ! Define ENABLE arglist offsets
195 0258 2 ! Offset in ENB_ARGS_ADR of no. of enable args following
196 0259 2 ! Addr. of longword containing
197 0260 2 ! UNWIND action code.
198 0261 2 ! Addr. of OPEN/CLOSE canonical array  
0262 2  
199 0263 2 MAP  
200 0264 2 ! SIG_ARGS_ADR : REF BLOCK [, BYTE]. | SIGNAL args
201 0265 2 ! MCH_ARGS_ADR : REF BLOCK [, BYTE]. | mechanism args
202 0266 2 ! ENB_ARGS_ADR : REF VECTOR [OPECLO_ADR, LONG]; !ENABLE args list array  
0267 2  
203 0268 2 LOCAL  
204 0269 2 ! EST_FP : REF BLOCK [, BYTE], ! Establisher's FP
205 0270 2 ! SIG_PC_LOC: REF VECTOR [, LONG], ! Location of user PC in signal list
206 0271 2 ! OPECLO_ARRAY : REF VECTOR [OPEN$K_KEY_MAX + 1, LONG]; ! OPEN/CLOSE canonical array  
0272 2  
207 0273 2 !+ If this is unwind condition, perform cleanup. since
208 0274 2 ! Perform LUB cleanup indicated by EBABLE arg UNWIND_ACT_ADR
209 0275 2 ! (set by the establisher).
210 0276 2 !-  
211 0277 2  
212 0278 2 IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STSSV_COND_ID;, BYTE] EQL (SS$_UNWIND^-3)
213 0279 2 THEN  
214 0280 2 ! BEGIN
215 0281 2 ! CLEANUP_LUB(..ENB_ARGS_ADR [UNWIND_ACT_ADR]);
216 0282 2 ! RETURN SSS_NORMAL;
217 0283 2 ! END;  
218 0284 2  
219 0285 2 OPECLO_ARRAY = .ENB_ARGS_ADR [OPECLO_ADR];  
220 0286 2  
221 0287 2 !+ If this is not a FOR$ error or if another RTL handler has seen this
222 0288 2 ! error (noted by signal argument for user PC being non-zero) then
223 0289 2 ! just resignal.
224 0290 2 !-  
225 0291 2  
226 0292 2 IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STSSV_FAC_NO;, BYTE] NEQ FOR$K_FAC_NO
227 0293 2 THEN  
228 0294 2 ! RETURN SSS_RESIGNAL;  
229 0295 2  
230 0296 2 SIG_PC_LOC = SIG_ARGS_ADR [CHF$L_SIG_ARG1] + (.SIG_ARGS_ADR [CHF$L_SIG_ARG1] * %UPVAL);
231 0297 2  
232 0298 2 IF .SIG_PC_LOC [0] NEQ 0
233 0299 2 THEN
```

```
; 237 0300 2      RETURN SSS_RESIGNAL;
; 238 0301
; 239 0302
; 240 0303      + Check if user provided ERR= keyword or not. If yes, convert signal to
; 241 0304      a return to the caller of the establisher with condition value in R0.
; 242 0305      If IOSTAT is present, act as if ERR= is also.
; 243 0306      If caller omitted OPECLO_ADR entry in ENB_ARGS_ADR, treat as if no ERR=.
; 244 0307
; 245 0308
; 246 0309      IF .ENB_ARGS_ADR [ENABLE_COUNT] GEQU OPECLO_ADR AND (.OPECLO_ARRAY [OPEN$K_ERR] OR .OPECLO_ARRAY [
; 247 0310          OPEN$K_IOSTAT]) NEQ 0
; 248 0311      THEN
; 249 0312          BEGIN
; 250 0313
; 251 0314      + If IOSTAT was specified, store the value.
; 252 0315      - 
; 253 0316
; 254 0317
; 255 0318      IF .OPECLO_ARRAY [OPEN$K_IOSTAT] NEQ 0
; 256 0319      THEN
; 257 0320          BEGIN
; 258 0321
; 259 0322          LOCAL
; 260 0323              IOSTAT;
; 261 0324
; 262 0325              IOSTAT = .BLOCK [SIG_ARGS_ADR [CHFSL_SIG_NAME], STSSV_CODE:, BYTE];
; 263 0326
; 264 0327      IF .OPECLO_ARRAY [OPEN$K_IOSTAT_L]
; 265 0328      THEN
; 266 0329          .OPECLO_ARRAY [OPEN$K_IOSTAT] = .IOSTAT
; 267 0330      ELSE
; 268 0331          BEGIN
; 269 0332
; 270 0333          LOCAL
; 271 0334              IOSTAT_ADR : REF BLOCK [, BYTE];
; 272 0335
; 273 0336              IOSTAT_ADR = .OPECLO_ARRAY [OPEN$K_IOSTAT];
; 274 0337              IOSTAT_ADR [0, 0, 16, 0] = .IOSTAT;
; 275 0338
; 276 0339
; 277 0340
; 278 0341
; 279 0342      IF NOT CALLG (.AP, LIB$SIG_TO_RET) THEN FOR$$SIG_FATINT ()
; 280 0343
; 281 0344      END
; 282 0345      ELSE
; 283 0346
; 284 0347      + No ERR=, so set user call PC saved in stack frame of establisher and RESIGNAL
; 285 0348      - 
; 286 0349
; 287 0350
; 288 0351      BEGIN
; 289 0352          EST_FP = .MCH_ARGS_ADR [CHFSL_MCH_FRAME];
; 290 0353          SIG_PC_LOC [0] = .EST_FP [SF$C_SAVE_PC];
; 291 0354
; 292 0355
; 293 0356      END;                                ! End no ERR=
```

```

294 0357 2
295 0358 2      + Return resignal condition (ignored if SYSSUNWIND called).
296 0359 2      -
297 0360 2
298 0361 2      RETURN SSS_RESIGNAL
299 0362 1      END;

```

! End of FOR\$ERR\_OPECLO handler

.TITLE FOR\$ERROR Internal FORTRAN error handling module

.IDENT \1-022\

.EXTRN FORSSCB\_GET, FORSSCB\_POP  
.EXTRN FORSSFP\_MATCH, FORSSFREE\_VM  
.EXTRN FORSSCLOSE\_FILE  
.EXTRN FORSSSIG\_FATINT  
.EXTRN FORSSSIG\_DATCOR  
.EXTRN LIBSSIG\_TO\_RET

.PSECT \_FORSCODE,NOWRT, SHR, PIC.2

00000124	8F	04	A2	52	04	000C 00000	.ENTRY FOR\$ERR_OPECLO, Save R2,R3 MOVL SIG_ARGS_ADR, R2 CMPZV #3, #25, 4(R2), #292 BNEQ 1\$	0186
				19	03	ED 00006		0278
18	06	A2	50	0C	AC 00012	MOVL ENB_ARGS_ADR, R0	0281	
			00000V	04	B0 DD 00016	PUSHL 04(R0)	0282	
52	04	CF	50	01	FB 00019	CALLS #1, CLEANUP_LUB	0285	
			01	00	DO 0001E	MOVL #1, R0	0293	
53	0C	0C	53	0C	AC 00022	RET	0297	
			50	08	A3 00026	1\$: MOVL ENB_ARGS_ADR, R3	0298	
53	0C	A0	0C	00	ED 0002A	MOVL 8(R3), OPECLO_ARRAY	0309	
			51	08	A2 00030	CMPZV #0, #12, 6(R2), #24	0310	
52	04	A2	51	08	A241 DE 00032	BNEQ 5\$	0318	
			51	08	A241 DE 00036	MOVAL 8(R2)[R1], SIG_PC_LOC	0325	
53	0C	A0	02	61	D5 0003B	TSTL (SIG_PC_LOC)	0327	
			53	08	A2 0003D	BNEQ 5\$	0329	
53	0C	A0	53	63	D1 0003F	CMPL (R3), #2	0336	
			53	08	A0 C9 00044	BLSSU 4\$	0337	
53	0C	A0	53	58	2E 13 0004A	BISL3 88(OPECLO_ARRAY), 12(OPECLO_ARRAY), R3	0342	
			53	08	A0 D0 0004C	BEQL 4\$	0352	
53	0C	A0	53	15	13 00050	MOVL 88(OPECLO_ARRAY), R3	0353	
			53	08	A0 E9 00052	BEQL 3\$	0353	
52	04	A2	05	64	03 EF 00058	EXTZV #3, #12, 4(R2), IOSTAT	0353	
			63	05	A0 E9 00058	BLBC 100(OPECLO_ARRAY), 2\$	0353	
50	0C	A0	63	52	D0 0005C	MOVL IOSTAT, (R3)	0353	
			50	05	A0 E9 0005F	BRB 3\$	0353	
50	0C	A0	50	53	DO 00061	MOVL R3, IOSTAT_ADR	0353	
			50	05	A0 E9 00064	MOVW IOSTAT, (IOSTAT_ADR)	0353	
00000000G	00	00	50	6C	FA 00067	CALLG (AP), LIBSSIG_TO_RET	0353	
			15	50	E8 0006E	BLBS R0, 5\$	0353	
00000000G	00	00	50	00	FB 00071	CALLS #0, FORSSSIG_FATINT	0353	
			50	08	AC 00078	BRB 5\$	0353	
50	0C	A0	50	04	AC DO 0007A	MOVL MCH_ARGS_ADR, R0	0353	
			61	10	A0 DO 0007E	MOVL 4(R0), EST_FP	0353	
					61	MOVL 16(EST_FP), (SIG_PC_LOC)	0353	

FOR\$ERROR  
1-022

internal FORTRAN error handling module

F 5  
16-Sep-1984 00:20:31 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:31:54 [FORRTL.SRC]FORERROR.B32;1

Page 8  
(3)

50 0918 8F 3C 00086 5\$: MOVZWL #2328, R0  
04 0008B RET

; 0361  
; 0362

; Routine Size: 140 bytes. Routine Base: \_FOR\$CODE + 0000

; 300 0363 1

```

302 0364 1 GLOBAL ROUTINE FOR$ERR_ENDHND (
303 0365 1   SIG_ARGS_ADR,
304 0366 1   MCH_ARGS_ADR,
305 0367 1   ENB_ARGS_ADR)
306 0368 1   =
307 0369 1
308 0370 1   ++
309 0371 1   FUNCTIONAL DESCRIPTION:
310 0372 1
311 0373 1   FOR$ERR_ENDHND is an error condition handler established
312 0374 1   by each I/O statement which has an ERR= and END= error transfer
313 0375 1   mechanism (as an option of the user program).
314 0376 1
315 0377 1   If the signaled condition is FOR$_ENDDURREA (24="END-OF FILE DURING READ")
316 0378 1   and an END= has been specified by the user in his I/O statement
317 0379 1   (.END_EQL_ADR NEQ 0), the handler unwinds to the user specified address (by calling
318 0380 1   SYSSUNWIND with depth equal to CHFSL_MCH_DEPTH + ..INCR_DEPTH_ADR + 1)
319 0381 1   and new_pc equal to ..END_EQL_ADR.
320 0382 1   Otherwise, if an ERR= had been specified by the user in his I/O statement
321 0383 1   (ERR_EQL NEQ 0), the handler unwinds to the user specified address
322 0384 1   by calling SYSSUNWIND with depth equal to CHFSL_MCH_DEPTH + ..INCR_DEPTH_ADR + 1
323 0385 1   and new_pc equal to ..ERR_EQL_ADR.
324 0386 1
325 0387 1   If neither of the above cases holds, the error is ressignaled
326 0388 1   so that a user handler or the OTS default handler will get invoked.
327 0389 1   If UNWIND occurs, the appropriate cleanup takes place
328 0390 1   as indicated by the establisher in the ENABLE arg UNWIND_ACT_ADR.
329 0391 1   If FORSK_UNWINDPOP is indicated, the current LUB/ISB/RAB is popped.
330 0392 1   If FORSK_UNWINDRET is indicated, the LUB/ISB/RAB is returned and the
331 0393 1   file closed.
332 0394 1   Otherwise (FORSK_UNWINDNOP) nothing is done.
333 0395 1
334 0396 1   FORMAL PARAMETERS:
335 0397 1
336 0398 1   SIG_ARGS_ADR.ml.ra      Adr. of signal arg list
337 0399 1   MCH_ARGS_ADR.ml.ra      Adr. of mechanism arg list
338 0400 1   ENB_ARGS_ADR.ml.ra      Adr. of ENABLE arg list which contains:
339 0401 1   UNWIND_ACT_ADR.rl.r      Adr. of longword containing UNWIND action code.
340 0402 1   Any of FORSK_UNWINDNOP, FORSK_UNWINDPOP,
341 0403 1   FORSK_UNWINDRET.
342 0404 1   ERR_EQL_ADR.ra.r       Adr. of longword containing Adr. of the user address
343 0405 1   to be transferred to or 0 on any error condition
344 0406 1   END_EQL_ADR.ra.r       Adr. of longword containing Adr. of the user address
345 0407 1   to be transferred to or 0 on end-of-file
346 0408 1   INCR_DEPTH_ADR.rl.r     Adr. of longword containing Incremental no. of frames between the establishe
347 0409 1   and the users program (usually 0 or 1).
348 0410 1   Note: All parameters to a condition handler must be addresses of values in BLISS if used in an ENABLE.
349 0411 1
350 0412 1   IMPLICIT INPUTS:
351 0413 1
352 0414 1   FORSSA_CUR_LUB          Adr. of current LUB/ISB/RAB or 0
353 0415 1   Note: obtained by calling FORSSCB_GET rather than directly.
354 0416 1
355 0417 1   IMPLICIT OUTPUTS:
356 0418 1
357 0419 1   SIG_ARGS_ADR[SIGS_USER_PC] Set to user call PC to RTL
358 0420 1

```

```

359 0421 1 : COMPLETION CODES:
360 0422 1
361 0423 1 : SSS_RESIGNAL if no ERR= or END= was specified by user, so that
362 0424 1 : a user handler or the default OTS handler will get a chance.
363 0425 1 : SSS_NORMAL if unwind called (although ignored if unwind called)
364 0426 1
365 0427 1 : SIDE EFFECTS:
366 0428 1
367 0429 1 : If END= and EOF OR ERR= was specified, the stack is unwound
368 0430 1 : to user and new_PC is set from ..END_EQL_ADR or .ERR_EQL_ADR.
369 0431 1 : If unwind, the current LUB/ISB/RAB may be popped or returned.
370 0432 1 --
371 0433 1
372 0434 2 : BEGIN
373 0435 2
374 0436 2
375 0437 2 : LOCAL
376 0438 2 :   EST_FP : REF BLOCK [ , BYTE],           ! Establisher's FP
377 0439 2 :   SIG_PC_LOC: REF VECTOR [ , LONG];  ! Location of user PC in signal list
378 0440 2 : LITERAL
379 0441 2 :   UNWIND_ACT_ADR = 1,                  ! Declare offsets in ENABLE VECTOR arg list
380 0442 2 :   ERR_EQ_ADR = 2,                     ! UNWIND action code
381 0443 2 :   END_EQL_ADR = 3,                   ! ERR= adr or 0
382 0444 2 :   INCR_DEPTH_ADR = 4;                ! END= adr or 0
383 0445 2 :                                         ! incremental depth
384 0446 2
385 0447 2 : MAP
386 0448 2 :   SIG_ARGS_ADR : REF BLOCK [ , BYTE], ! SIGNAL arg list
387 0449 2 :   MCH_ARGS_ADR : REF BLOCK [ , BYTE], ! mechanism arg list
388 0450 2 :   ENB_ARGS_ADR : REF VECTOR [INCR_DEPTH_ADR + 1, LONG]; ! ENABLE arg list
389 0451 2
390 0452 2
391 0453 2 : +
392 0454 2 : | Check for unwinding since handler gets called when it does an unwind.
393 0455 2 : | If unwind, perform cleanup indicated by ENABLE arg UNWIND_ACT_ADR.
394 0456 2 : | Then return to the unwinder to keep unwinding (return value ignored).
395 0457 2 : -
396 0458 2 : IF .BLOCK [SIG_ARGS_ADR [CHFSL_SIG_NAME], STSSV_COND_ID:, BYTE] EQ (SSS_UNWIND^3)
397 0459 2 : THEN
398 0460 2 : BEGIN
399 0461 2 :   CLEANUP_LUB (..ENB_ARGS_ADR [UNWIND_ACT_ADR]);
400 0462 2 :   RETURN SSS_NORMAL;
401 0463 2 : END;
402 0464 2
403 0465 2 : +
404 0466 2 : | If error is not a FOR$ error or if another RTL handler has seen
405 0467 2 : | this error then resignal.
406 0468 2 : -
407 0469 2
408 0470 2 : IF .BLOCK [SIG_ARGS_ADR [CHFSL_SIG_NAME], STSSV_FAC_NO:, BYTE] NEQ FORSK_FAC_NO
409 0471 2 : THEN
410 0472 2 :   RETURN SSS_RESIGNAL;
411 0473 2
412 0474 2 : SIG_PC_LOC = SIG_ARGS_ADR [CHFSL_SIG_ARG1] + (.SIG_ARGS_ADR [CHFSL_SIG_ARG1] * %UPVAL);
413 0475 2 : IF .SIG_PC_LOC [0] NEQ 0
414 0476 2 : THEN
415 0477 2 :   RETURN SSS_RESIGNAL;

```

```
416      0478 2
417      0479 2
418      0480 2
419      0481 2
420      0482 2
421      0483 2
422      0484 2
423      0485 2
424      0486 2
425      0487 2
426      0488 2
427      0489 2
428      0490 2
429      0491 2
430      0492 2
431      0493 2
432      0494 2
433      0495 2
434      0496 2
435      0497 2
436      0498 2
437      0499 2
438      0500 2
439      0501 2
440      0502 2
441      0503 2
442      0504 2
443      0505 2
444      0506 2
445      0507 2
446      0508 2
447      0509 2
448      0510 2
449      0511 2
450      0512 2
451      0513 2
452      0514 2
453      0515 2
454      0516 2
455      0517 2
456      0518 2
457      0519 2
458      0520 2
459      0521 2
460      0522 2
461      0523 2
462      0524 2
463      0525 2
464      0526 2
465      0527 2
466      0528 2
467      0529 2
468      0530 2
469      0531 2
470      0532 2
471      0533 2
472      0534 2

;+ Check for END= and ERR=.
;+ If this is end-of-file (during read)
;+ Unwind to the user with the new_pc being .END_ADR and with
;+ R0 as an IOSTAT value of -1.
;-
;+
;+ IF ..ENB_ARGS_ADR [END_EQL_ADR] NEQA 0 AND .SIG_ARGS_ADR [CHF$L_SIG_NAME] EQL FORS_ENDDURREA
;+ THEN
;+ BEGIN
;+ LOCAL
;+   T;
;+   MCH_ARGS_ADR [CHF$L_MCH_SAVR0] = -1;
;+   T = .MCH_ARGS_ADR [CHF$C_MCH_DEPTH] + ..ENB_ARGS_ADR [INCR_DEPTH_ADR] + 1;
;+   IF $UNWIND (DEPADR = T, NEWPC = ..ENB_ARGS_ADR [END_EQL_ADR])
;+   THEN
;+     RETURN SSS_NORMAL
;+   ELSE
;+     FOR$SIG_FATINT ()
;+ END;
;-
;+ If this is an error, and ERR= was specified by the user.
;+ Unwind to the user with the new_pc being .ERR_ADR and
;+ with R0 set to the proper IOSTAT value.
;-
;+
;+ IF ..ENB_ARGS_ADR [ERR_EQL_ADR] NEQA 0
;+ THEN
;+ BEGIN
;+ LOCAL
;+   T;
;+   IF .SIG_ARGS_ADR [CHF$L_SIG_NAME] EQL FORS_ENDDURREA
;+   THEN
;+     MCH_ARGS_ADR [CHF$L_MCH_SAVR0] = -1
;+   ELSE
;+     MCH_ARGS_ADR [CHF$L_MCH_SAVR0] = .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STSSV_CODE:, BYTE];
;+   T = .MCH_ARGS_ADR [CHF$L_MCH_DEPTH] + ..ENB_ARGS_ADR [INCR_DEPTH_ADR] + 1;
;+   IF $UNWIND (DEPADR = T, NEWPC = ..ENB_ARGS_ADR [ERR_EQL_ADR])
;+   THEN
;+     RETURN SSS_NORMAL
;+   ELSE
;+     FOR$SIG_FATINT ()
;+ END;
;-
;+ If neither END= nor ERR= specified by user.
```

·EXTRN SYSSUNWIND

00000124	BF	63	56	00000000G	00	007C	00000	.ENTRY	FORSSERR_ENDHND, Save R2,R3,R4,R5,R6	0364
			55	00000000G	00	9E	00002	MOVAB	FORSSSIG_FATINT, R6	
			5E		08	C2	00010	MOVAB	SYSSUNWIND, R5	
			52	04	AC	00	00013	SUBL2	#8, SP	
			53	04	A2	9E	00017	MOVL	SIG_ARGS_ADR, R2	0458
			19		03	ED	0001B	MOVAB	4(R2), R3	
					0F	12	00024	CMPZV	#3, #25, (R3), #292	
			50	0C	AC	00	00026	BNEQ	1S	
				04	B0	DD	0002A	MOVL	ENB_ARGS_ADR, R0	0461
			0000V	CF		01	FB	PUSHL	24(R0)	
					0080	31	00032	CALLS	#1, CLEANUP_LUB	
18	02	A3	0C		00	ED	00035	BRW	7S	0462
					08	12	0003B	1\$: CMPZV	#0, #12, 2(R3), #24	0470
			50	08	A2	00	0003D	BNEQ	2S	0474
			54	08	A240	DE	00041	MOVL	8(R2), R0	
					64	D5	00046	MOVAL	8(R2)[R0], SIG_PC_LOC	0475
					03	13	00048	TSTL	(SIG_PC_LOC)	
					0089	31	0004A	BEQL	3S	
			52	0C	AC	00	0004D	BRW	12S	0486
				0C	B2	D5	00051	1\$: MOVL	ENB_ARGS_ADR, R2	
					2A	13	00054	TSTL	212(R2)	
			001880C4	8F		63	D1	BEQL	4S	
					21	12	00056	CMPL	(R3), #1605828	
			50	08	50	08	AC	BNEQ	4S	0493
				0C	A0	01	DD	MOVL	MCH_ARGS_ADR, R0	
			50	08	A0	01	CE	MNEG	#1-12(R0)	
				0C	A0	82	C1	ADDL	216(R2), 8(R0), R0	0494
				6E	01	A0	9E	MOVAB	1(R0), f	
					01	B2	DD	PUSHL	212(R2)	0496
					0C	B2	00071	PUSHAB	T	
					04	AE	9F	CALLS	#2, SYSSUNWIND	
					65	02	FB	BLBS	R0, 7S	
					38	50	E8	CALLS	#0, FORSSSIG_FATINT	0500
					66	00	FB	TSTL	28(R2)	0510
					08	B2	D5	BEQL	9S	
			001880C4	BF	08	37	13	MOVL	MCH_ARGS_ADR, R0	0519
					50	AC	00083	CMPL	(R3), #1605828	0517
					63	D1	00089			

FOR\$ERROR  
1-022

Internal FORTRAN error handling module

K 5  
16-Sep-1984 00:20:31  
14-Sep-1984 12:31:54

VAX-11 Bliss-32 v4.0-742  
[FORRTL.SRC]FORERROR.B32;1

Page 13  
(4)

FOR  
1-0

OC	A0	06	12	00090	BNEQ	5\$		0519	
63	50	01	CE	00092	MNEG	#1, 12(R0)			
OC	A0	06	11	00096	BRB	6\$			
50	08	03	EF	00098	5\$:	EXTZV	#3, #12, (R3), 12(R0)	0521	
04	AE	10	B2	C1	0009E	ADDL3	@16(R2), 8(R0), R0	0523	
		01	AO	9E	000A4	MOVAB	1(R0)		
		08	B2	DD	000A9	PUSHL	@8(R2)	0525	
		08	AE	9F	000AC	PUSHAB	T		
		65	02	FB	000AF	CALLS	#2, SYSSUNWIND		
		04	50	E9	000B2	BLBC	R0, 8\$		
		50	01	DO	000B5	7\$:	MOVL	#1, R0	0527
				04	000B8	RET			
		66	00	FB	000B9	8\$:	CALLS	#0, FOR\$SIG_FATINT	0529
		50	08	AC	000BC	9\$:	MOVL	MCH_ARGS_ADR, R0	0541
		50	04	AO	000C0	MOVL	4(R0), EST_FP		
51	10	B2	01	C1	000C4	ADDL3	#1, @16(R2), I	0543	
			04	11	000C9	BRB	11\$		
		50	0C	A0	DO 000CB	10\$:	MOVL	12(EST_FP), EST_FP	0544
		F9	51	F5	000CF	11\$:	SOBGTR	I, 10\$	
		64	10	A0	DO 000D2		MOVL	16(EST_FP), (SIG_PC_LOC)	0546
		50	0918	8F	3C 000D6	12\$:	MOVZWL	#2328, R0	0548
				04	000DB	RET		0549	

: Routine Size: 220 bytes. Routine Base: \_FOR\$CODE + 008C

: 488 0550 1

```

490 0551 1 GLOBAL ROUTINE FOR$IOSTAT_HND (
491 0552 1     SIG_ARGS_ADR,
492 0553 1     MCH_ARGS_ADR,
493 0554 1     ENB_ARGS_ADR)
494 0555 1     =
495 0556 1
496 0557 1     ++
497 0558 1     FUNCTIONAL DESCRIPTION:
498 0559 1
499 0560 1     FOR$IOSTAT_HND is an error condition handler established by each
500 0561 1     auxilliary I/O statement which can have as optional arguments
501 0562 1     ERR= and IOSTAT=.
502 0563 1
503 0564 1     If the enable argument ERR_EQL_ADR is non zero, FOR$IOSTAT_HND
504 0565 1     unwinds with the saved R0 set to the appropriate IOSTAT small
505 0566 1     integer FORTRAN error number. If ERR_EQL_ADR is zero, then it
506 0567 1     is assumed that no ERR= is present and the error is ressignalled.
507 0568 1     Note that the unwind is not done to the ERR= address, rather the
508 0569 1     compiled code makes a test of the returned value and branches
509 0570 1     to the designated ERR= statement itself.
510 0571 1
511 0572 1     If UNWIND occurs, the appropriate cleanup takes place,
512 0573 1     as indicated by the establisher in the ENABLE arg UNWIND_ACT_ADR.
513 0574 1     If FORSK_UNWINDPOP is indicated, the current LUB/ISB/RAB is popped.
514 0575 1     If FORSK_UNWINDRET is indicated, the LUB/ISB/RAB is returned and the
515 0576 1     file closed.
516 0577 1     Otherwise (FORSK_UNWINDNOP) nothing is done.
517 0578 1
518 0579 1     FORMAL PARAMETERS:
519 0580 1
520 0581 1     SIG_ARGS_ADR.ml.ra      Adr. of signal arg list
521 0582 1     MCH_ARGS_ADR.ml.ra    Adr. of mechanism arg list
522 0583 1     ENB_ARGS_ADR.ml.ra    Adr. of ENABLE arg list which contains:
523 0584 1         UNWIND_ACT_ADR.rl.r  Adr. of longword containing UNWIND action code.
524 0585 1         Any of FORSK_UNWINDNOP, FORSK_UNWINDPOP,
525 0586 1         FORSK_UNWINDRET.
526 0587 1     ERR_EQL_ADR.rl.v      0 if there is no ERR= on the statement
527 0588 1                  1 if there is an ERR= present.
528 0589 1     Note: All parameters to a condition handler must be addresses of values in BLISS if used in an ENABLE.
529 0590 1
530 0591 1     IMPLICIT INPUTS:
531 0592 1
532 0593 1     FORSSA_CUR_LUB        Adr. of current LUB/ISB/RAB or 0
533 0594 1
534 0595 1     Note: obtained by calling FORSSCB_GET rather than directly.
535 0596 1
536 0597 1
537 0598 1     MCH_ARGS_ADR [CHFSL_MCH_SAVR0] Set to an IOSTAT value
538 0599 1
539 0600 1     COMPLETION CODES:
540 0601 1
541 0602 1     SSS_RESIGNAL if no ERR= or END= was specified by user, so that
542 0603 1     a user handler or the default OTS handler will get a chance.
543 0604 1     SSS_NORMAL if unwind called (although ignored if unwind called)
544 0605 1
545 0606 1     SIDE EFFECTS:
546 0607 1

```

```

547 0608 1 | If ERR= was specified, the stack is unwound to the user.
548 0609 1 | If unwind, the current LUB/ISB/RAB may be popped or returned.
549 0610 1 |-- BEGIN
550 0611 2
551 0612 2 | LOCAL
552 0613 2 |   EST_FPC : REF BLOCK [ , BYTE],           ! Establisher's FP
553 0614 2 |   SIG_PC_LOC: REF VECTOR [ , LONG];    ! Location of user PC in signal list
554 0615 2 | LITERAL
555 0616 2 |   UNWIND_ACT_ADR = 1,                  ! Declare offsets in ENABLE VECTOR arg list
556 0617 2 |   ERR_EQ[ADR] = 2;                      ! UNWIND action code
557 0618 2 |   ERR_EQ[ADR] = 2;                      ! ERR= present, 1 or 0
558 0619 2
559 0620 2
560 0621 2
561 0622 2
562 0623 2 | MAP
563 0624 2 |   SIG_ARGS_ADR : REF BLOCK [ , BYTE],   ! SIGNAL arg list
564 0625 2 |   MCH_ARGS_ADR : REF BLOCK [ , BYTE],   ! mechanism arg list
565 0626 2 |   ENB_ARGS_ADR : REF VECTOR [ERR_EQ[ADR] + 1, LONG]; ! ENABLE arg list
566 0627 2
567 0628 2 |+
568 0629 2 |   Check for unwinding since handler gets called when it does an unwind.
569 0630 2 |   If unwind, perform cleanup indicated by ENABLE arg UNWIND_ACT_ADR.
570 0631 2 |   Then return to the unwinder to keep unwinding (return value ignored).
571 0632 2 |-
572 0633 2 | IF .BLOCK [SIG_ARGS_ADR [CHFSL_SIG_NAME], STSSV_COND_ID:, BYTE] EQ (SSS_UNWIND^3)
573 0634 2 | THEN
574 0635 2 |   BEGIN
575 0636 2 |   CLEANUP_LUB(..ENB_ARGS_ADR [UNWIND_ACT_ADR]);
576 0637 2 |   RETURN SSS_NORMAL;
577 0638 2 | END;
578 0639 2
579 0640 2 |+
580 0641 2 |   If this is not a FORS error or if another RTL handler has seen this
581 0642 2 |   error (noted by signal argument for user PC being non-zero) then
582 0643 2 |   just resignal.
583 0644 2 |-
584 0645 2
585 0646 2 | IF .BLOCK [SIG_ARGS_ADR [CHFSL_SIG_NAME], STSSV_FAC_NO:, BYTE] NEQ FORSK_FAC_NO
586 0647 2 | THEN
587 0648 2 |   RETURN SSS_RESIGNAL;
588 0649 2 |   SIG_PC_LOC = SIG_ARGS_ADR [CHFSL_SIG_ARG1] + (.SIG_ARGS_ADR [CHFSL_SIG_ARG1] * XUPVAL);
589 0650 2 |   IF .SIG_PC_LOC [0] NEQ 0
590 0651 2 |   THEN
591 0652 2 |   RETURN SSS_RESIGNAL;
592 0653 2
593 0654 2 |+
594 0655 2 |   If this is an error, and ERR= was specified by the user.
595 0656 2 |   Unwind to the user with saved R0 being the IOSTAT value.
596 0657 2 |-
597 0658 2
598 0659 2 | IF ..ENB_ARGS_ADR [ERR_EQ[ADR]] NEQA 0
599 0660 2 | THEN
600 0661 2 |   BEGIN
601 0662 2 |   MCH_ARGS_ADR [CHFSL_MCH_SAVR0] = .BLOCK [SIG_ARGS_ADR [CHFSL_SIG_NAME], STSSV_CODE:, BYTE];
602 0663 2
603 0664 2 | IF $UNWIND ()

```

```

: 604      0665 3      THEN
: 605      0666 3      RETURN SSS_NORMAL
: 606      0667 3      ELSE FOR$SIG_FATINT ();
: 607      0668 2      END;
: 608      0669 2
: 609      0670 2
: 610      0671 2
: 611      0672 2
: 612      0673 2      !+ If ERR= not specified by the user
: 613      0674 2      scan back from frame of establisher to frame of routine to called by user.
: 614      0675 2      Set user CALL PC to library in SIGNAL arg list.
: 615      0676 2      Just indicate to the condition handling facility to resignal the condition
: 616      0677 2      so that a user supplied handler or the OTS default handler will get a chance to handle.
: 617      0678 2
: 618      0679 2
: 619      0680 2      EST_FP = .MCH_ARGS_ADR [CHFSL MCH FRAME];
: 620      0681 2      SIG_PC_LOC [0] = .EST_FP [SF$[_SAVE_PC]];
: 621      0682 2      RETURN SSS_RESIGNAL
: 622      0683 1      END;                                !End of FOR$IOSTAT_HND

```

00000124	8F	04	A2	52	04	000C 00000	.ENTRY FOR\$IOSTAT_HND, Save R2,R3	: 0551
				19	03	00002	MOVL SIG_ARGS_ADR, R2	: 0633
					0E	00006	CMPZV #3, #25, -4(R2), #292	
				50	AC	00010	BNEQ 1\$	
				04	DD	00012	MOVL ENB_ARGS_ADR, R0	: 0636
				01	00016	PUSHL #4(R0)		
				35	FB	00019	CALLS #1, CLEANUP_LUB	
				35	11	0001E	BRB 2\$	
				00	ED	00020	1\$: CMPZV #0, #12, 6(R2), #24	: 0637
				44	12	00026	BNEQ 5\$	: 0646
				53	A2	00028	MOVL 8(R2), R0	: 0649
				08	DE	0002C	MOVAL 8(R2)[R0], SIG_PC_LOC	: 0650
				63	D5	00031	TSTL (SIG_PC_LOC)	
				37	12	00033	BNEQ 5\$	
				50	AC	00035	MOVL ENB_ARGS_ADR, R0	: 0659
				08	BO	00039	TSTL #8(R0)	
				22	13	0003C	BEQL 4\$	
				50	AC	0003E	MOVL MCH_ARGS_ADR, R0	: 0662
				03	EF	00042	EXTZV #3, #12, -4(R2), 12(R0)	
				7E	7C	00049	CLRQ -(SP)	: 0664
				00	FB	0004B	CALLS #2, SYSSUNWIND	
				04	E9	00052	BLBC R0, 3\$	
				50	01	00055	MOVL #1, R0	: 0666
					04	00058	RET	
				00	FB	00059	CALLS #0, FOR\$SIG_FATINT	: 0668
				50	AC	00060	MOVL MCH_ARGS_ADR, R0	: 0680
				50	04	A0	MOVL 4(R0), EST_FP	
				63	10	A0	MOVL 16(EST_FP), (SIG_PC_LOC)	: 0681
				50	0918	8F	MOVZWL #2328, R0	: 0682
					3C	0006C		
					04	00071	RET	: 0683

: Routine Size: 114 bytes, Routine Base: \_FORSCODE + 0168

FOR\$ERROR  
1-022

Internal FORTRAN error handling module

8 6  
16-Sep-1984 00:20:31  
14-Sep-1984 12:31:54  
VAX-11 Bliss-32 v4.0-742  
[FORRTL.SRC]FORERROR.B32;1

Page 17  
(5)

: 623

0684 1

FOR  
1-0

```

625 0685 1 GLOBAL ROUTINE FOR$IO_IN_PROG {
626 0686 1     SIG_ARGS_ADDR
627 0687 1     MCH_ARGS_ADDRS
628 0688 1     =
629 0689 1
630 0690 1     ++
631 0691 1     FUNCTIONAL DESCRIPTION:
632 0692 1
633 0693 1     FOR$IO_IN_PROG is a special handler that is designed to
634 0694 1     allow the Run-Time Library to clean I/O that is in progress
635 0695 1     when an error occurs during the processing of a multi-call
636 0696 1     I/O statement. For example, if evaluation of a variable
637 0697 1     list item in a WRITE statement causes an error to be signalled,
638 0698 1     there is no RTL handler in the stack frame to catch the error
639 0699 1     and clean up in the case of an unwind
640 0700 1
641 0701 1     This handler is enabled at the user's stack frame level. The
642 0702 1     address of whatever user handler that was in the frame is stored
643 0703 1     in the ISB. When an error is signalled, this handler finds
644 0704 1     the address of the user handler, if any, and calls it. There
645 0705 1     should be no normally detectable difference caused by FOR$IO_IN_PROG
646 0706 1     being on the frame. On unwind, the current ISB is popped and the
647 0707 1     user's handler is called again. This way, we are protected against
648 0708 1     all errors on all call levels.
649 0709 1
650 0710 1     FORMAL PARAMETERS:
651 0711 1
652 0712 1     SIG_ARGS_ADDR.ml.ra      Address of signal arguments list
653 0713 1     MCH_ARGS_ADDR.ml.ra    Address of mechanism arguments list
654 0714 1
655 0715 1     IMPLICIT INPUTS:
656 0716 1
657 0717 1     ISB/LUB/RAB database
658 0718 1
659 0719 1     IMPLICIT OUTPUTS:
660 0720 1
661 0721 1     ISB/LUB/RAB database
662 0722 1
663 0723 1     COMPLETION CODES:
664 0724 1
665 0725 1     Whatever is returned by the user handler.
666 0726 1     --
667 0727 1
668 0728 2     BEGIN
669 0729 2
670 0730 2     GLOBAL REGISTER
671 0731 2     CCB = 11 : REF $FOR$CCB_DECL;
672 0732 2
673 0733 2
674 0734 2     BUILTIN
675 0735 2     CALLG,
676 0736 2     AP;
677 0737 2
678 0738 2     LOCAL
679 0739 2     USER_HANDLER,
680 0740 2     EST_FP : REF BLOCK [. BYTE];
681 0741 2     MAP

```

: R

```

682 0742 2 SIG_ARGS_ADR : REF BLOCK [, BYTE];      | signal argument list
683 0743 2 MCH_ARGS_ADR : REF BLOCK [, BYTE];      | mechanism argument list
684 0744 2
685 0745 2
686 0746 2
687 0747 2
688 0748 2
689 0749 2 EST_FP = .MCH_ARGS_ADR [CHF$L_MCH_FRAME];
690 0750 2
691 0751 2
692 0752 2
693 0753 2
694 0754 2
695 0755 2 IF .BLOCK [SIG_ARGS_ADR [CHF$L_SIG_NAME], STSSV_COND_ID:, BYTE] EQL (SSS_UNWIND^3)
696 0756 2 THEN
697 0757 2   BEGIN
698 0758 2     FORSSCB_GET ();                      ! Get address of current LUB
699 0759 2
700 0760 2     IF .EST_FP NEQ .CCB [ISBSA_USER_FP] THEN FORSSSIG_FATINT ();    ! Error
701 0761 2
702 0762 2     USER_HANDLER = .CCB [ISBSA_USR_HANDL];  ! Get user's handler address
703 0763 2     CLEARUP_LUB (FOR$K_UNWINDPOP);          ! Clean up LUB and restore user's handler
704 0764 2
705 0765 2     IF .USER_HANDLER NEQ 0 THEN RETURN CALLG (.AP, .USER_HANDLER);
706 0766 2
707 0767 2     RETURN SSS_NORMAL;
708 0768 2     END;
709 0769 2
710 0770 2
711 0771 2
712 0772 2
713 0773 2
714 0774 2
715 0775 2
716 0776 2
717 0777 2
718 0778 2
719 0779 2
720 0780 2
721 0781 2
722 0782 2
723 0783 2
724 0784 2
725 0785 2
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
170
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
172
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
173
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
174
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
175
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
176
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
177
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
178
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
179
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
180
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
181
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
182
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
183
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
184
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
185
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
186
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
187
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
188
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
189
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
190
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
191
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
192
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
193
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
194
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
195
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
196
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
197
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
198
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
199
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
200
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
201
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
202
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
203
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
204
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
205
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
206
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
207
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
208
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
209
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
210
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
211
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
212
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
213
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
214
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
215
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
216
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
217
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
218
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
219
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
220
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
221
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
222
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
223
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
224
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
225
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
226
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
227
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
228
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
229
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
230
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
231
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
232
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
233
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
234
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
235
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
236
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
237
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
238
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
239
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
240
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
241
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
242
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
243
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
244
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
245
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
246
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
247
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
248
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
249
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
250
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
251
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
252
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
253
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
254
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
255
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
256
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
257
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
258
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
259
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
260
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
261
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
262
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
263
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
264
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
265
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
266
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
267
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
268
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
269
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
270
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
271
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
272
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
273
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
274
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
275
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
276
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
277
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
278
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
279
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
280
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
281
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
282
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
283
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
284
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
285
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
286
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
287
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
288
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
289
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
290
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
291
2920
2921
2922
2
```

00000124	8F	04	A0	50 53 50 19	08 04 04	080C AC A0 A0	00000 0002 00006 0000A	.ENTRY	FOR\$10 IN PROG, Save R2,R3,R11
						03	ED	MOVL	MCH ARG5 ADR, R0
						28	0000E	MOVL	4(R0), EST FP
						12	00018	MOVL	SIG ARG5 ADR, R0
						00	0001A	CMPZV	#3, #25, 4(R0), #292
						53	00020	BNEQ	28
				FF4C	CB	00000000G		JSB	FOR\$CB_GET
								CMPL	EST FP, -180(CCB)

FOR\$ERROR  
1-022

Internal FORTRAN error handling module

E 6  
16-Sep-1984 00:20:31  
14-Sep-1984 12:31:54 VAX-11 Bliss-32 V4.0-742  
[FORRTL.SRC]FORERROR.B32;1

Page 20  
(6)

FOR  
1-0

00000000G	00		07	13	00025		BEQL	1\$		
	52		00	FB	00027		CALLS	#0	FOR\$SIG_FATINT	
		FF44	CB	D0	0002E	1\$:	MOVL	-188(CC8),	USER_HANDLER	0762
0000V	CF		7E	D4	00033		CLRL	-(SP)		0763
			01	FB	00035		CALLS	#1,	CLEANUP_LUB	
			52	D5	0003A		TSTL	USER_HANDLER		0765
			14	12	0003C		BNEQ	3\$		
	50		01	D0	0003E		MOVL	#1,	RO	0767
				04	00041		RET			
00000000G	00		53	DD	00042	2\$:	PUSHL	EST_FP		0775
	52		01	FB	00044		CALLS	#1	-FOR\$FP_MATCH	
		FF44	CB	D0	0004B		MOVL	-188(CC8),	USER_HANDLER	0781
	62		04	13	00050		BEQL	4\$		0783
			6C	FA	00052	3\$:	CALLG	(AP),	(USER_HANDLER)	
	50	0918	8F	3C	00056	4\$:	RET			
				04	0005B		MOVZWL	#2328,	RO	
							RET			0785

: Routine Size: 92 bytes. Routine Base: \_FOR\$CODE + 01DA

: 726 0786 1

```
728 0787 1 ROUTINE CLEANUP_LUB (ACTION) : NOVALUE =
729 0788 1
730 0789 1 ++ FUNCTIONAL DESCRIPTION:
731 0790 1 Perform the UNWIND action indicated by ACTION on the current LUB.
732 0791 1
733 0792 1 FORMAL PARAMETERS:
734 0793 1
735 0794 1 ACTION.rlu.v FOR$K_UNWINDNOP, FOR$K_UNWINDPOP, or FOR$K_UNWINDRET.
736 0795 1
737 0796 1 --+
738 0797 1
739 0798 1 BEGIN
740 0800 2
741 0801 2 GLOBAL REGISTER
742 0802 2 CCB = 11 : REF $FOR$CCB_DECL;
743 0803 2
744 0804 2 BIND
745 0805 2 FAB = CCB: REF $FOR$FAB_CCB_STRUCT;
746 0806 2
747 0807 2 CASE .ACTION FROM FOR$K_UNWINDPOP TO FOR$K_UNWINDRET OF
748 0808 2 SET
749 0809 2
750 0810 2
751 0811 2
752 0812 2
753 0813 2
754 0814 2
755 0815 2
756 0816 2
757 0817 2
758 0818 2
759 0819 2
760 0820 2
761 0821 2
762 0822 2
763 0823 2
764 0824 2
765 0825 2
766 0826 2
767 0827 2
768 0828 2
769 0829 2
770 0830 2
771 0831 2
772 0832 2
773 0833 2
774 0834 2
775 0835 2
776 0836 2
777 0837 2
778 0838 2
779 0839 2
780 0840 2
781 0841 2
782 0842 2
783 0843 2
784 0844 2

: R

+ If the UNWIND action is to pop the LUB/ISB/RAB, call CB_POP to do
the work.
!-
[FOR$K_UNWINDPOP] :
BEGIN
LOCAL
USER_FP; ! User's FP
FOR$CB_GET ();
USER_FP = .CCB [ISBSA_USER_FP]; ! CCB set to adr. of current /LUB/ISB/RAB
! Get user's FP
IF .USER_FP NEQ 0 THEN .USER_FP = .CCB [ISBSA_USR_HANDL]; ! Restore user's handler
CCB [RABSL_UBF] = .CCB [LUBSA_RBUF_ADR];
CCB [RABSW_USZ] = .CCB [LUBSW_RBUF_SIZE];
FOR$CB_POP ();
END;

+ If the UNWIND action is NOP, do nothing.
!-
[FOR$K_UNWINDNOP] :
:
+ If the UNWIND action is RET, then try to $CLOSE the file associated
with this LUB/ISB/RAB. Deallocate any dynamic storage associated
with this LUB. Return the LUB/ISB/RAB to free storage.
```

```

785 0844 2      [FOR$K_UNWINDRET] :
786 0845      BEGIN
787 0846      FOR$SCB_GET ();
788 0847      ! Set CCB to adr. of current LUB/ISB/RAB
789 0848      |+ See if file is RMS opened.
790 0849      |-
791 0850
792 0851      IF (.FAB [FABSW_IFI] NEQ 0)
793 0852      THEN
794 0853      |+
795 0854      Do an RMS Close of the file, and arrange to deallocate its LUB/ISB/RAB
796 0855      when all I/O to it is finished. Normally, we are doing the only I/O
797 0856      to it.
798 0857      |-
799 0858      FOR$CLOSE_FILE ()
800 0859      ELSE
801 0860      |+
802 0861      Even though the file is not open, we wish to deallocate the LUB, since
803 0862      this is the simplest way to reinitialize it if the user tries to use
804 0863      the logical unit number again, so tell OTSS$POP_CCB to deallocate it.
805 0864      |-
806 0865      CCB [LUB$V DEALLOC] = 1;
807 0866
808 0867      |+
809 0868      We are done with the logical unit.
810 0869      |-
811 0870      FOR$SCB_POP ();
812 0871      END;
813 0872      TES;
814 0873      END;
815 0874      END;

```

0804 00000 CLEANUP_LUB:											
02	0020	52	0000000G	00	9E	00002	WORD	Save R2,R11		0787	
		00		04	AC	00009	MOVAB	FOR\$SCB_GET, R2		0808	
		003A		0006		0000E	CASEL	ACTION, #0, #2			
						1\$:	.WORD	2S-1\$,-			
								7S-1\$,-			
								4S-1\$			
		50	FF4C	62	16	00014	2\$:	JSB	FOR\$SCB_GET	0822	
				CB	D0	00016		MOVL	-180(CCB), USER_FP	0823	
				05	13	0001B		BEQL	3\$	0825	
		24	60	FF44	CB	D0	0001D	MOVL	-188(CCB), (USER FP)	0827	
			AB	EC	AB	D0	00022	3\$:	MOVL	-20(CCB), 36(CCB)	0828
		20	AB	D2	AB	B0	00027	MOVW	-46(CCB), 32(CCB)	0829	
						14	11	BRB	6\$	0846	
						62	16	JSB	FOR\$SCB_GET	0851	
						62	0002E	TSTW	70(FAB)		
						46	AB	BEQL	5\$		
						09	13	CALLS	#0, FOR\$CLOSE_FILE	0858	
						00	FB	BRB	6\$		
						04	11	BISB2	#16 -1(FAB)	0865	
						FF	AB	JSB	FOR\$SCB_POP	0870	
						0000000G	00				
							16				
						000042	6\$:				

FOR\$ERROR  
1-022

Internal FORTRAN error handling module

H 6  
16-Sep-1984 00:20:31  
14-Sep-1984 12:31:54

VAX-11 Bliss-32 V4.0-742  
[FORRTL.SRC]FORERROR.B32;1

Page 23  
(7)

: 0874

: Routine Size: 73 bytes, Routine Base: \_FOR\$CODE + 0236

: 816 0875 1  
: 817 0876 1 END  
: 818 0877 1  
: 819 0878 0 ELUDOM

: !End of module

#### PSECT SUMMARY

Name	Bytes	Attributes
_FOR\$CODE	639	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

#### Library Statistics

File	Total	Symbols	Pages	Processing
	Total	Loaded	Mapped	Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	18	581	00:01.0
\$255\$DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	190	52	00:00.6
\$255\$DUA28:[FORRTL.OBJ]RTLLIB.L32;1	36	0	8	00:00.1

#### COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS:FORERROR/OBJ=OBJ\$:FORERROR MSRC\$:FORERROR/UPDATE=(ENH\$:FORERROR)

: Size: 639 code + 0 data bytes  
: Run Time: 00:16.9  
: Elapsed Time: 00:45.4  
: Lines/CPU Min: 3120  
: Lexemes/CPU-Min: 15828  
: Memory Used: 119 pages  
: Compilation Complete

FOR  
1-0

S  
R  
E  
L  
M  
C

0180 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

